A minimal Statechart Virtual Machine

Principal Investigator: Dr. Nicolas F. Rouquette

Mission Execution and Automation Section
Jet Propulsion Laboratory
Mail Stop: 301-270
4800 Oak Grove Drive
Pasadena, California 91109
Voice: 818-354-9600 FAX: 818-393-6004
Nicolas.F.Rouquette@jpl.nasa.gov

Co-Investigators:

Mary M. Lam	**Patrick L. Crouse**
Autonomy and Control Section Jet Propulsion Laboratory Mail Stop : 301-350 4800 Oak Grove Drive Pasadena, California 91109 Voice: 818-354-3675 FAX: 818-393-3654 Email: mailto:Mary.M.Lam@jpl.nasa.gov	NASA Goddard Space Flight Center Mail Stop: 581.1 8800 Greenbelt Road Greenbelt, Maryland 20771 Voice: 301-286-9613 FAX: 301-286-0243 Email: mailto:Patrick.L.Crouse@gsfc.nasa.gov

1 Problem Statement

To leverage the full potential of Statecharts as a highly intuitive and expressive diagrammatic notation of behavior, we need a virtual machine capability for dynamically managing Statecharts in a running system: a Statechart Virtual Machine. This capability should provide a low-cost solution for Statechart management, specifically, one that does not rely on making expensive flight software modifications. Furthermore, it should be fully accountable in the following sense: first, explaining the behavior of a Statechart throughout its execution, and second, supporting the analytic verification of formal properties across all possible executions of a Statechart.

The principal investigator gathered a team of scientists – Prof. David Harel at the Weizmann Institute of Science in Israel --technologists at NASA Ames (Dr. Klaus Havelund), NASA Goddard Space Flight Center (Patrick Crouse) and other consultants to build a Statechart Virtual Machine as part of a research proposal to NASA UPN 632. One component of that proposal includes a flight experiment on the WIRE testbed platform. Patrick Crouse, a SMEX mission director at Goddard is unsure of the WIRE availability beyond FY00. Thus, there is a strong incentive to develop the Statechart Virtual Machine experiment for WIRE as early as possible to avoid loosing a flight experiment opportunity.

2 Justification

The UPN 632 proposal represents a challenge to coordinate scientists and technologies spread across the US and the world. The effort is substantial, it requires over $1M of funding for over 1.5 work years for 3 years. An early prototype would help considerably this effort by identifying the difficult issues to work on.

Although Statecharts have been highly successful for designing the fault protection capability of Deep Space One, we don't have an adequate characterization of the resources required to deploy a virtual machine for Statecharts in a resource-limited environment. What's the minimum level of CPU performance needed? Clearly, the 35 Mips RAD 6000 has proven sufficient to execute the compiled Statecharts of Deep Space One's fault protection system. It is also adequate for a much larger system like the Remote Agent Experiment at 0.1 Hz. Would 10 Mips suffice for a Statechart Virtual Machine of some kind? Performance characterization helps but if the 386 processor of the WIRE testbed can do it, then we will have a valuable reference point. The minimal Statechart Virtual Machine is deliberately reduced in scope so that the performance envelope of the 386 processor does not constitute an insurmountable constraint.

3 Benefits

The main benefit of building the minimal Statechart Virtual Machine is to take advantage of the WIRE testbed availability while it lasts. There are a number of other indirect benefits as well. As part of the UPN 632 proposal, we have identified a broad range of potential applications for a Statechart Virtual Machine. A fairly large number of people would like to see this concept materialize to better evaluate it. Bob Rasmussen also encouraged me to aggressively seek sources of funding to do just that.

A flight demonstration on WIRE will force working out technology transfer and deployment issues between JPL and GSFC. This difficulty should then make it easier to coordinate similar

applications of the STVM to internal projects at JPL, where conventional wisdom often blurs the distinction between technology transfer and reuse with people reuse.

As a low-power, resource-tight computing platform, WIRE is an excellent testbed for project managers who contemplate using Statecharts in rovers and other small devices. For those who contemplate using Statecharts across the board for all behavioral aspects, the flight experiment will give a better metric for evaluating how well the technology scales up.

Most importantly, the benefit of this minimal Statechart Virtual Machine is to exercise and prototype early the technologies and tools involved in the UPN 632 proposal for the full-scale Statechart Virtual Machine. This is particularly important to help us organize the project in two sensible deliverables relative to the Statechart notation and semantics, for example, finite state machines at end of the first year and full Statechart notation and semantics at the end of the second year.

Finally, this proposal gives us an opportunity to study both the technologies involved in building a Statechart Virtual machine capability, including those pertaining to ground systems operations.

4 What is a Statechart Virtual Machine

The core of a Statechart Virtual Machine (STVM) involves four main components: an XML parser, a scheduler and modules for introspection and monitoring (See Figure 1). The UML specification includes an XML interchange format for UML model content. This makes working with UML models in XML a vendor-neutral, tool-neutral process and saves tool-specific interfaces and code generators.

4.1 Statechart Execution Scheduler

The scheduler is the core Statechart execution engine. The Statechart semantics defined in the UML specification leave a number of open decisions, such as the order of resolution for conflicting transitions and the interleaving of concurrent AND states. For this proposal, such issues are drastically simplified because there is no concurrency in simple states and simple transitions. For full Statechart notation and semantics, this scheduler provides explicit access to the scheduling mechanisms and policies that govern Statechart execution. This access is particularly important to interface model-based checking tools like those developed in Dr. Michael Lowry's group at NASA Ames.

4.2 Statechart Introspection and Monitoring

Introspection refers to the process of querying the Statechart database about the static definition of a Statechart characteristics and possible behavior. Monitoring refers to the process of querying the scheduler about the dynamic properties of a Statechart execution behavior. Introspection is not necessary to demonstrate the Minimal STVM capability. Monitoring is kept at a minimum to produce a log of states and transitions visited during execution.

4.3 Statechart Virtual Machine Architecture

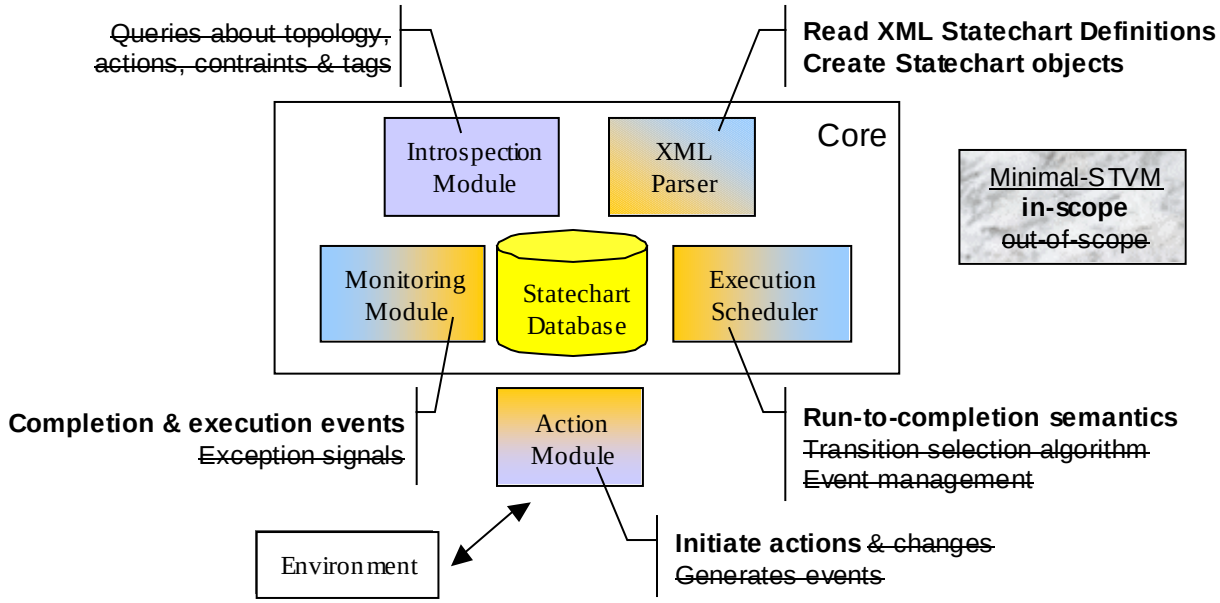

Figure 1: Scope & Architecture of Minimal Statechart Virtual Machine

5 Schedule and Deliverables

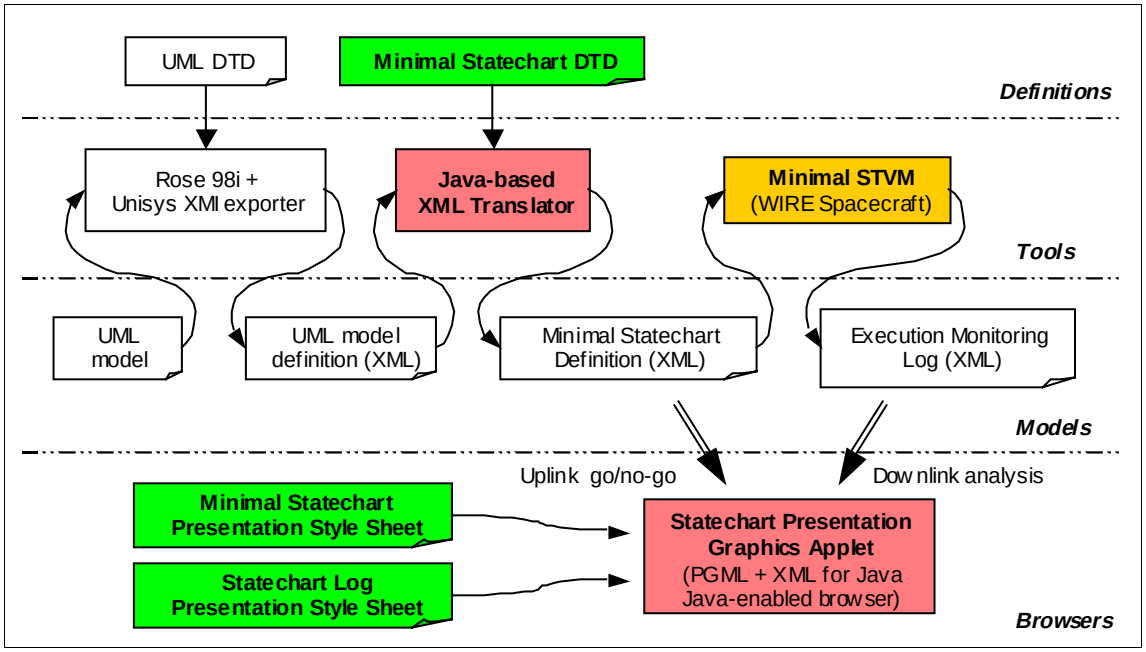

Figure 2: Minimal Statechart Virtual Machine and related workflow products

The deliverable products involved in this proposal are shown in Figure 2 (bold items); they are described in the table below.

Item	*Description*
Minimal Statechart DTD	A subset of the UML DTD restricted to describe Statecharts made of simple states and simple transitions
XML Translator	A Java application to apply the Minimal Statechart DTD for extracting simple Statecharts from models
Minimal STVM (XML Parser)	A trimmed version of James Clark's XML parser http://www.jclark.com/xml/expat.html
Minimal STVM (Execution Scheduler, Monitoring & Action Modules, Statechart Database)	Software designed with Rhapsody for C. Full-scale code generation from Rhapsody model to C. Software developed according to WIRE documentation & processes http://tracedata.nascom.nasa.gov/~wire/ up to readiness for flight experiement.
Minimal Statechart Definition & Log Style Sheets & Presentation Graphics	A Java applet based on the Portable Graphics Markup Language (PGML) for rendering Statecharts & log annotations in a Java-enabled browser

This proposal involves three classes of personnel differentiated by their skills base:

- "ground" software activities pertaining to processing UML models, extracting Minimal Statecharts with the XML-based tools and visualizing Minimal Statechart definitions and execution logs. The kills involved include: Java and XML-related technologies.
- "flight" software activities pertaining to designing the Minimal STVM with Rhapsody for C, with a special emphasis towards full-scale automatic code generation.
- "integration & test" software activities pertaining to embedding the Minimal Statechart Virtual Machine first in a unit test environment for WIRE and second in a ground-based WIRE testbed at Goddard.

Because funding for WIRE operations follows fiscal year cycles, the deliverables described here are to be completed within FY00. The minimal criteria of successful completion consists in demonstrating the Minimal STVM in a ground-based unit test environment either at JPL or at Goddard. The workflow products identified in Figure 2 will also be streamlined and automated according to the automatic code generation technology work of Dr. Nicolas Rouquette originally proposed to NASA IV&V to be completed under a CSMISS-funded extension through 3/1/2000 (See http://eis.jpl.nasa.gov/auto_sw).

6 Cost, Work Plan & Technical Approach

The principal investigator and co-investigators are planning to work with two full-time junior staff. This proposal involves a lot of emerging software engineering technologies and practices that unbiased junior staff can pick up quickly with proper mentoring from senior staff. The technical depth of the proposal is intentionally kept at a minimum to allow the junior staff to emphasize rigor with respect to applicable software engineering practices and policies. The funding requested for this proposal is $60k.

7 Proposal evaluation

7.1 Return on investment

This proposal represents almost all of the first year UPN 632 $50k activity towards defining the WIRE experiment. The UPN 632 version includes hierarchical states to the Minimal STVM described here; a trivial extension to make.

Both Deep Impact (DI) and Mars Sample Return (MSR) want to reuse the Deep Space One Fault-Protection architecture. However, the DS1 FP engine is, at the core, a mechanism for executing compiled Statecharts. It would be a simple task to take the CSMISS Minimal STVM in C as a replacement of the DS1 FP compiled Statechart core. This would advance the technology transfer to DI and MSR, thereby doubling the CSMISS investment. For DI, the return is further increased with the flexibility of the STVM versus compiled Statecharts. This can save a full time system engineer easily 25% time on early prototyping and increased test flexibility. At current JPL labor rates, that's roughly $40k of burdened costs.

The total estimated return on investment after would thus be: $50k + $120k + $40k = $210k.

7.2 Endorsement

This proposal is the lite version of a UPN 632 proposal. There has not been sufficient time to reach the co-investigators of the UPN 632 proposal to receive their endorsements. These co-investigators are:

- Mary Lam from JPL
- Prof. David Harel, Dean of the Faculty of Mathematics at the Weizmann Institute and Chief Technologist at I-Logix
- Dr. Klaus Havelund from NASA ARC
- Patrick Crouse, SMEX mission director at NASA GSFC

7.3 Participation with Industry and Academia

Prof. David Harel will study safety issues related to modifying Statecharts in a running system as part of the UPN 632 work. In this proposal, such criteria are relatively trivial to state. Dr. Bruce Douglass from I-Logix has consulted the PI regarding applying the ROPES process. We expect the Design Hub to continue the consulting contract with I-Logix throughout the duration of this proposal. This is important because we have never applied the ROPES process on any completed project yet.